This specification provides a set of algorithms to consistently generate colors given a string. The string can be a nickname, a JID or any other piece of information. All entities adhering to this specification generate the same color for the same string, which provides a consistent user experience across platforms.
Legal

Copyright

This XMPP Extension Protocol is copyright © 1999 – 2018 by the XMPP Standards Foundation (XSF).

Permissions

Permission is hereby granted, free of charge, to any person obtaining a copy of this specification (the "Specification"), to make use of the Specification without restriction, including without limitation the rights to implement the Specification in a software program, deploy the Specification in a network service, and copy, modify, merge, publish, translate, distribute, sublicense, or sell copies of the Specification, and to permit persons to whom the Specification is furnished to do so, subject to the condition that the foregoing copyright notice and this permission notice shall be included in all copies or substantial portions of the Specification. Unless separate permission is granted, modified works that are redistributed shall not contain misleading information regarding the authors, title, number, or publisher of the Specification, and shall not claim endorsement of the modified works by the authors, any organization or project to which the authors belong, or the XMPP Standards Foundation.

Warranty

## NOTE WELL: This Specification is provided on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. ##

Liability

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall the XMPP Standards Foundation or any author of this Specification be liable for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising from, out of, or in connection with the Specification or the implementation, deployment, or other use of the Specification (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if the XMPP Standards Foundation or such author has been advised of the possibility of such damages.

Conformance

This XMPP Extension Protocol has been contributed in full conformance with the XSF’s Intellectual Property Rights Policy (a copy of which can be found at <https://xmpp.org/about/xsf/ipr-policy> or obtained by writing to XMPP Standards Foundation, P.O. Box 787, Parker, CO 80134 USA).
13 Test Vectors

13.1 Test Vectors ......................................................... 10

13.1.1 No Color Vision Deficiency correction .......................... 10
13.1.2 With Red/Green-blindness correction ............................ 10
13.1.3 With Blue-blindness correction .................................. 10

13.2 Test Vectors for mapping to 216 color palette ....................... 10

13.2.1 No Color Vision Deficiency correction .......................... 11
13.2.2 With Red/Green-blindness correction ............................ 11
13.2.3 With Blue-blindness correction ................................. 11
1 Introduction

Colors provide a valuable visual cue to recognize shapes. Recognition of colors works much faster than recognition of text. Together with the length and overall shape of a piece of text (such as a nickname), a color provides a decent amount of entropy to distinguish a reasonable amount of entities, without having to actually read the text.

Clients have been using randomly or deterministically chosen colors for users in multi-user situations for a long time already. However, since there has been no standard for how this is implemented, the experience differs across platforms. The goal of this XEP is to provide a uniform, platform-independent, stateless and easy-to-implement way to map arbitrary bytestrings to colors, as well as give recommendations how this is applied to color names of participants in conversations, roster entries and other pieces of text.

To allow cross-client use, it is important that the color scheme can be adapted to different environments. This specification provides means to adapt colors to different background colors as well as Color Vision Deficiencies.

In no way is the system presented in this specification a replacement for names. It only serves as an additional visual aid.

2 Requirements

The color generation mechanism should provide the following features:

- Consistent generation of color across all platforms depending solely on the identifier used as input for the algorithm.

- The system should be reasonably fast; it must be possible to, for example, apply it to all roster entries even of very large rosters in reasonable amount of time.

- It must be able to provide decent contrast on any background.

- The implementation should be stateless and not be complex.

- A fallback path for users with common types of Color Vision Deficiencies must be provided.

- A fallback path for systems which can only use colors from a restricted palette must be provided.

3 Use Cases

3.1 Generating a color

To generate a color from a string of text, the following algorithms are applied in order:
1. Generate a Hue value from the text.

2. If enabled, apply configured corrections for Color Vision Deficiencies.

3. If the output device only supports a small palette of colors, map the angle to the closest palette color.

4. If the output device supports RGB output, Convert the angle to a RGB.

3.2 Adding colors to participants of a conversation

Implementations may colorize the participants of a conversation with an individual color to make them easier to distinguish.
In such cases, the color SHOULD be generated as described in the Generating a color section.
The input used SHOULD be, in descending order of preference, (a) the bare JID of the user (not the room), (b) the nickname as chosen by the user in the room.

3.3 Auto-Generating Avatars

Implementations may want to show a picture in connection with a contact even if the contact does not have an avatar defined (e.g. via User Avatar (XEP-0084) ¹).
In such cases, auto-generating an avatar SHOULD happen as follows:

1. Obtain a name for the contact, in descending order of preference, (a) the nickname from the conversation, (b) the bare JID of the contact (not the bare JID of the conference in case of a Multi-User Chat (XEP-0045) ² room).

2. Generate a color as described in the Generating a color section.

3. Fill an implementation-defined background shape with that color.

4. Render the first character of the name in white or black centered on the shape.

4 Business Rules

• Implementations SHOULD allow the user to turn off any colorization completely.

• Implementations SHOULD implement the Color Vision Deficiency profiles and SHOULD allow the user to choose any of these profiles or to disable the correction.

• Implementations MUST NOT share the Color Vision Deficiency correction settings with other entities.

5 Algorithms

The algorithms in this document use the HSLuv \(^3\) color space. It provides consistent brightness (for a given luminosity) across its entire definition space. There is also widespread library support.

5.1 Angle generation

Input: An identifier, encoded as octets of UTF-8 (RFC 3269 \(^4\)).
Output: Hue angle.
Note: The goal of this algorithm is to convert arbitrary text into a scalar value which can then be used to calculate a color.

1. Run the input through SHA-1 (RFC 3174 \(^5\)).
2. Treat the output as little endian and extract the least-significant 16 bits. (These are the first two bytes of the output, with the second byte being the most significant one.)
3. Divide the value by 65536 (use float division) and multiply it by 360 (to map it to degrees in a full circle).

5.2 Corrections for Color Vision Deficiencies

Input: Hue angle.
Output: Hue angle.
Note: This algorithm will re-map the angle to map it away from ranges which can not be distinguished by people with the respective Color Vision Deficiencies.
Note: Some floating-point modulo implementations will return negative outputs for negative inputs. This algorithm assumes that your implementation returns non-negative outputs for all inputs.

5.2.1 Red/Green-blindness

Add 90 to the angle, take it modulo 180 and subtract 90. Take the result modulo 360 to ensure that it’s in the range from 0 to 360.
Note: the same effect can be achieved by forcing the two most-significant bits of the angle to be equal to the second-most-significant bit before converting to a float in Angle generation. This avoids having to perform a floating-point modulo operation.

\(^3\)HSLuv <http://www.hsluv.org/>.
5.2.2 Blue-blindness

Take the angle modulo 180.
Note: the same effect can be achieved by setting the most-significant bit to zero before conversion to floating point in Angle generation. This avoids having to perform a floating-point modulo operation.

5.3 Adapting the Color for specific Background Colors

Input: RGB values for the color to adapt (Ri, Gi, Bi) and for the background color to adapt to (Rb, Gb, Bb), in the range from 0 to 1 each.
Output: Values for Red (Rc), Green (Gc) and Blue (Bc) in the RGB color space in the range from 0 to 1.

1. Invert the background color by subtracting the individual channels from 1 each:
   \[
   \begin{align*}
   rb_{\text{inv}} &= 1 - rb; \\
   gb_{\text{inv}} &= 1 - gb; \\
   bb_{\text{inv}} &= 1 - bb;
   \end{align*}
   \]

2. Mix the inverted background with the color to adapt, using a mixing factor of 0.2:
   \[
   \begin{align*}
   rc &= 0.2 \times rb_{\text{inv}} + 0.8 \times ri; \\
   gc &= 0.2 \times gb_{\text{inv}} + 0.8 \times gi; \\
   bc &= 0.2 \times bb_{\text{inv}} + 0.8 \times bi;
   \end{align*}
   \]

5.4 RGB generation

Use the HSLuv operation hsluvToRgb to convert the Hue angle to a color. For this, saturation SHOULD be set to 100 and lightness SHOULD be set to 50.

5.5 Conversion of an RGB color palette to a Hue palette

Input: A set of RGB colors (each component from 0 to 1).
Output: A mapping from angles (integer, from 0 to 360) to RGB colors.
Note: when the algorithm finishes, the mapping maps angles (rounded to two decimal places) to the R, G, B triples which come closest to the desired color and lightness.

1. Create an empty mapping M which maps from Hue angles to quadruples of L, R, G and B.
2. For each color R, G, B from the input palette:
   a) If the R, G and B values are equal, skip the color and continue with the next. (Grayscale does not work well, since its saturation and hue are undefined.)
6 IMPLEMENTATION NOTES


c) Round the angle to the next integer value.

d) If the angle is not in the mapping M yet, or if the L value of the existing entry is further away from 73.2 than the new L value, put the L, R, G, and B values as value for the angle into the mapping.

3. Strip the L values from the values of mapping M.

4. Return M as the result of the algorithm.

Implementations are free to choose a representation for palette colors different from R, G, B triplets. The exact representation does not matter, as long as it can be converted to a Hue angle accordingly.

5.6 Mapping of a Hue angle to closest palette color

Input: (a) A mapping which maps angles to R, G, B triplets and (b) a color to map to the closest palette color as angle alpha.

Output: A palette color as R, G, B triplet.

Note: See Conversion of an RGB color palette to a Hue palette on how to convert an R, G, B triplet to an angle.

1. First, check if alpha rounded to an integer. If so, return that match immediately.

2. For each angle beta in the palette, calculate the distance metric:

   \[ D = \min((\alpha - \beta) \mod 360, (\beta - \alpha) \mod 360) \]

3. Return the R, G, B triplet associated with the angle with the smallest distance metric D.

Implementations are free to choose a representation for palette colors different from R, G, B triplets. The exact representation does not matter, as long as it can be converted to a Hue angle accordingly.

6 Implementation Notes

6.1 Gamma Correction

Implementations should be aware of Gamma correction and apply it as needed.
6.2 Background Color Correction

An implementation which shows the generated colors on a colored background SHOULD apply Adapting the Color for specific Background Colors. If the background is not uniformly colored, it is up to the implementation to determine an appropriate surrogate background color to correct against.

If an implementation shows the generated colors on a grayscale (including white and black) background, it MAY apply the background color correction algorithm. It is RECOMMENDED to always apply the algorithm if the background color is changed dynamically, to avoid discontinuities between grayscale and colored backgrounds.

Implementations SHOULD use the same background color for all generated colors. If this is not feasible, implementations SHOULD use the same background color for all generated colors within the same GUI control (for example, within a conversation and within the roster).

6.3 Normalization

When processing JIDs as text input, implementations MUST prepare the JID as it would for comparing it to another JID with a case-sensitive comparison function.

7 Accessibility Considerations

As outlined above, implementations SHOULD offer the Red/Green-Blindness and Blue-Blindness corrections as defined in the Corrections for Color Vision Deficiencies section. Users SHOULD be allowed to choose between:

- disabling all corrections (skip the Corrections for Color Vision Deficiencies step entirely),
- applying one of the Color Vision Deficiency correction profiles and
- disabling colorization altogether.

The last option is useful for users with monochromatic view or who find colors distracting.

Some sources on the internet indicate that people with Color Vision Deficiencies may profit from having larger areas of color to be able to recognize them. This should be taken into consideration when selecting font weights and line widths for colored parts.

8 Security Considerations

This specification extracts a bit more information from an entity and shows it alongside the existing information to the user. As the algorithm is likely to produce different colors for look-alikes (see Best Practices to Prevent JID Mimicking (XEP-0165)\(^6\) for examples) in JIDs, it

may add additional protection against attacks based on those. Due to the limited set of distinguishable colors and only extracting 16 bits of the hash function output, possible Color Vision Deficiencies and/or use of palettes, entities MUST NOT rely on colors being unique in any context.

9 Design Considerations

This section provides an overview of design considerations made while writing this specification. It shows alternatives which have been considered, and eventually rejected.

9.1 The YCbCr color space

The versions up to 0.5 of this document used a variant of the YCbCr color space (namely BT.601) along with a custom algorithm to convert from angles to CbCr and from there to RGB. The HSLuv color space provides extremely consistent apparent brightness of the colors which cannot be achieved with simple application of YCbCr. In addition, HSLuv has widespread library support.

9.2 Hue-Saturation-Value/Lightness color space

The HSV and HSL color spaces fail to provide uniform luminosity with fixed value/lightness and saturation parameters. Adapting those parameters for uniform luminosity across the hue range would have complicated the algorithm with little to no gain.

9.3 Palette-based and context-aware coloring

Given a fixed-size and finite palette of colors, it would be possible to ensure that, until the number of entities to color exceeds the number of colors, no color collisions happen. There are issues with this approach when the set of entities is dynamic. In such cases, it is possible that an entity changes its associated color (for example by re-joining a colored group chat), which defeats the original purpose. In addition, more state needs to be taken into account, increasing the complexity of choosing a color.

9.4 Choice of mixing function in angle generation

This specification needs to collapse an arbitrarily long string into just a few bits (the angle in the CbCr plane). To do so, SHA-1 (RFC 3174) is used. CRC32 and Adler32 have been considered as faster alternatives. Downsides of these functions:

- Bad mixing without additional entropy.
- Adler32 is rarely available in standard libraries.
- CRC32 is ambiguous: there are multiple polynomials in widespread use (e.g. the Ethernet and the zlib polynomials). Often it is not clear which polynomial is used by a library.

SHA-1 is widely available. From a security point of view, the exact choice of hash function does not matter here, since it is truncated to 16 bits. At this length, any cryptographic hash function is weak.

9.5 Palette-mapping function

The palette-mapping algorithm operates on angles only and disregards the Y value except if the angles match. This has the downside that the brightness is not equal over the range of the palette mapped colors. The alternative would be to require Y to be close to the target Y. This has several issues:

- We cannot know if a palette can satisfy the given Y at all.
- Many colors from e.g. the "Web Safe" palette (used in 256 color terminals and the test vectors) will not satisfy any given Y, reducing the size of the effective palette drastically.

For the sake of having more colors available, the given algorithm was chosen which prefers many colors with hue conformance over fewer colors with hue and lightness conformance.

9.6 Input for color generation in a Multi-User Chat (XEP-0045) context

In Multi-User Chat (XEP-0045) conversations (MUCs), there are two viable choices for the hash function input when generating a color for a participant: the nickname as chosen by the participant (or their full JID) and the participants real bare JID. Both options have advantages and disadvantages. The advantages of using the nickname are:

- Yields the same output even in anonymous MUCs if the same nickname is used.
• Is guaranteed to work for transports implementing Multi-User Chat (XEP-0045) 10.

The advantages of using the bare JID are:

• Allows to use the same color in 1:1 communications with that entity as in group chats, helping with recognizability.
• Stays constant even across changes of ephemeral information like nicknames.

There is no obvious correct choice to make here; both choices break in different use-cases. Specifically, the "nickname" choice breaks when the same entity has different nicknames in different rooms, as well as when two different entities have the same nickname in different rooms. The "bare JID" choice breaks when (semi-)anonymous MUCs are involved. The choice "bare JID" has the conceptual advantage that it ties as closely as possible to the identity of the entity. It is also forward-compatible with future protocols where nicknames might not be available or work differently.

10 IANA Considerations

This document requires no interaction with the Internet Assigned Numbers Authority (IANA) 11.

11 XMPP Registrar Considerations

This document requires no interaction with the XMPP Registrar 12.

12 Acknowledgements

Thanks to Klaus Herberth, Daniel Gultsch, Georg Lukas, Tobias Markmann, Christian Schudt, and Marcus Waldvogel for their input and feedback on this document.

11 The Internet Assigned Numbers Authority (IANA) is the central coordinator for the assignment of unique parameter values for Internet protocols, such as port numbers and URI schemes. For further information, see <http://www.iana.org/>.
12 The XMPP Registrar maintains a list of reserved protocol namespaces as well as registries of parameters used in the context of XMPP extension protocols approved by the XMPP Standards Foundation. For further information, see <https://xmpp.org/registrar/>.
13 Test Vectors

13.1 Test Vectors

This section holds test vectors for the different configurations. The test vectors are provided as Comma Separated Values. Strings are enclosed by single quotes ('). The first line contains a header. Each row contains, in that order, the original text, the text encoded as UTF-8 as hexadecimal octets, the angle in degrees, the calculated hue in degrees (differs from angle only for CVD-corrected rows), and the Red, Green, and Blue values.

13.1.1 No Color Vision Deficiency correction

<table>
<thead>
<tr>
<th>text</th>
<th>hextext</th>
<th>angle</th>
<th>hue</th>
<th>r</th>
<th>g</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>526f6d656f</td>
<td>327.255249</td>
<td>327.255249</td>
<td>0.865</td>
<td>0.000</td>
<td>0.686</td>
</tr>
<tr>
<td><a href="mailto:juliet@capulet.lit">juliet@capulet.lit</a></td>
<td>6a756c69657440636170756c657426c6974</td>
<td>,</td>
<td>209.410400, 209.410400, 0.000, 0.515, 0.573</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'f09f98ba'</td>
<td>, 331.199341, 331.199341, 0.872, 0.000, 0.659</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>council</td>
<td>636f756e63696c</td>
<td>, 359.994507</td>
<td>359.994507</td>
<td>0.918</td>
<td>0.000</td>
<td>0.394</td>
</tr>
<tr>
<td>Board</td>
<td>426f617264</td>
<td>, 171.430664, 171.430664, 0.000, 0.527, 0.457</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.1.2 With Red/Green-blindness correction

<table>
<thead>
<tr>
<th>text</th>
<th>hextext</th>
<th>angle</th>
<th>hue</th>
<th>r</th>
<th>g</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>526f6d656f</td>
<td>327.255249</td>
<td>327.255249</td>
<td>0.865</td>
<td>0.000</td>
<td>0.686</td>
</tr>
<tr>
<td><a href="mailto:juliet@capulet.lit">juliet@capulet.lit</a></td>
<td>6a756c69657440636170756c657426c6974</td>
<td>,</td>
<td>209.410400, 29.410400, 0.742, 0.359, 0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'f09f98ba'</td>
<td>, 331.199341, 331.199341, 0.872, 0.000, 0.659</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>council</td>
<td>636f756e63696c</td>
<td>, 359.994507</td>
<td>359.994507</td>
<td>0.918</td>
<td>0.000</td>
<td>0.394</td>
</tr>
<tr>
<td>Board</td>
<td>426f617264</td>
<td>, 171.430664, 351.430664, 0.000, 0.494</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.1.3 With Blue-blindness correction

<table>
<thead>
<tr>
<th>text</th>
<th>hextext</th>
<th>angle</th>
<th>hue</th>
<th>r</th>
<th>g</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Romeo</td>
<td>526f6d656f</td>
<td>327.255249</td>
<td>147.255249</td>
<td>0.535</td>
<td>0.350</td>
<td></td>
</tr>
<tr>
<td><a href="mailto:juliet@capulet.lit">juliet@capulet.lit</a></td>
<td>6a756c69657440636170756c657426c6974</td>
<td>,</td>
<td>209.410400, 29.410400, 0.742, 0.359, 0.000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>'f09f98ba'</td>
<td>, 331.199341, 151.199341, 0.000, 0.533, 0.373</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>council</td>
<td>636f756e63696c</td>
<td>, 359.994507</td>
<td>179.994507</td>
<td>0.000</td>
<td>0.524</td>
<td>0.485</td>
</tr>
<tr>
<td>Board</td>
<td>426f617264</td>
<td>, 171.430664, 171.430664, 0.000, 0.527, 0.457</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13.2 Test Vectors for mapping to 216 color palette

The used palette can be generated by sampling the RGB cube evenly with six samples on each axis (resulting in 210 colors (grayscale is excluded)). The resulting palette is commonly
known as the palette of so-called "Web Safe" colors.

13.2.1 No Color Vision Deficiency correction

```
text, hextext, hue, best_hue, r, g, b
'Romeo', '526f6d656f', 327.255249, 327, 1.000, 0.000, 0.800
'juliet@capulet.lit', '6a756c69657440636170756c65742e6c6974',
  209.410400, 226, 0.000, 0.800, 1.000
'juliet@capulet.lit', 'f09f98ba', 331.199341, 331, 1.000, 0.400, 0.800
'council', '636f756e63696c', 359.994507, 359, 0.800, 0.200, 0.400
'Board', '426f617264', 171.430664, 161, 0.000, 1.000, 0.800
```

13.2.2 With Red/Green-blindness correction

```
text, hextext, hue, best_hue, r, g, b
'Romeo', '526f6d656f', 327.255249, 327, 1.000, 0.000, 0.800
'juliet@capulet.lit', '6a756c69657440636170756c65742e6c6974',
  29.410400, 30, 1.000, 0.600, 0.400
'juliet@capulet.lit', 'f09f98ba', 331.199341, 331, 1.000, 0.400, 0.800
'council', '636f756e63696c', 359.994507, 359, 0.800, 0.200, 0.400
'Board', '426f617264', 171.430664, 161, 0.000, 1.000, 0.800
```

13.2.3 With Blue-blindness correction

```
text, hextext, hue, best_hue, r, g, b
'Romeo', '526f6d656f', 147.255249, 147, 0.400, 0.800, 0.600
'juliet@capulet.lit', '6a756c69657440636170756c65742e6c6974',
  29.410400, 30, 1.000, 0.600, 0.400
'juliet@capulet.lit', 'f09f98ba', 151.199341, 153, 0.200, 0.800, 0.600
'council', '636f756e63696c', 179.994507, 192, 0.000, 0.800, 0.800
'Board', '426f617264', 171.430664, 161, 0.000, 1.000, 0.800
```